Java – Basic Syntax and Structure of Java Programs

Session Objectives

- Write and execute a simple Java program.
- Learn syntax rules and conventions.
- Explore identifiers, modifiers, and variables.

---

1. First Java Program

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

Steps to Compile and Run:

1. Save as `HelloWorld.java`
2. Compile: `javac HelloWorld.java`
3. Run: `java HelloWorld`

---

2. Java Basic Syntax

- Java is **case-sensitive** (`Hello` ≠ `hello`)
- Class names: Start with **uppercase**
- Method names: Start with **lowercase**
- File name = public class name + `.java`
- Every Java program starts with `public static void main(String[] args)`

---

3. Java Identifiers

Rules:

- Start with letter, _, or `$`
- Cannot use keywords
- Case-sensitive

☑ Valid: `name`, `_value`, `$amount`

✗ Invalid: `3score`, `void`

---

4. Java Modifiers

Java uses **access modifiers** like `public`, `private`, and `default`, and **non-access modifiers** like `static`, `final`, and `abstract`.

☑ Sample Codes for Modifiers

Access Modifiers

```
public class Person {         // public: accessible everywhere
    private String name;      // private: accessible only in this class

    void setName(String n) { // default: accessible in same package
        name = n;
    }

    public String getName() {
        return name;
    }
}
```

Non-Access Modifiers

```
public class Example {
    static int count = 0;          // static: belongs to the class, not objects
    final double PI = 3.14;        // final: value cannot be changed

    public static void main(String[] args) {
        System.out.println("Count: " + count);
        System.out.println("PI: " + new Example().PI);
    }
}
```

// `abstract` is used for abstract classes and methods (covered in advanced sessions)

---

5. Java Variables

Variables are used to **store data**.

☑ Real-Life Example – Student Info

```
public class StudentInfo {
    public static void main(String[] args) {
        String name = "Jessa";
        int age = 18;
        double grade = 92.5;

        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Grade: " + grade);
    }
}
```

📝 **Tip:** Use descriptive names for variables.

---

6. Java Arrays

Arrays store **multiple values** of the same type.

☑ Real-Life Example – Grocery List

```
public class GroceryList {
    public static void main(String[] args) {
        String[] items = {"Apples", "Bread", "Milk"};

        System.out.println("Grocery List:");
        System.out.println(items[0]);
        System.out.println(items[1]);
        System.out.println(items[2]);
    }
}
```

☑ Common Array Operations

Creating & Initializing Arrays

```
public class ArrayCreation {
    public static void main(String[] args) {
        // Method 1: Create and initialize with values
        int[] scores = {85, 92, 78, 96, 88};

        // Method 2: Create empty array, then assign values
        String[] subjects = new String[3];
        subjects[0] = "Math";
        subjects[1] = "Science";
        subjects[2] = "English";

        System.out.println("First score: " + scores[0]);
        System.out.println("First subject: " + subjects[0]);
    }
}
```

Getting Array Length

```
public class ArrayLength {
    public static void main(String[] args) {
        String[] colors = {"Red", "Blue", "Green", "Yellow"};

        System.out.println("Number of colors: " + colors.length);
        System.out.println("Last color: " + colors[colors.length - 1]);
    }
}
```

Reading and Writing Array Elements

```
public class ArrayReadWrite {
    public static void main(String[] args) {
        double[] prices = {12.99, 8.50, 15.75};

        // Reading from array
        System.out.println("Original first price: $" + prices[0]);

        // Writing to array (updating value)
        prices[0] = 11.99; // Update first price

        // Reading updated value
        System.out.println("Updated first price: $" + prices[0]);
        System.out.println("Second price: $" + prices[1]);
        System.out.println("Third price: $" + prices[2]);
    }
}
```

📝 **Tip:** Use index numbers to access each item (starting from 0).

---

7. Java Enums

Enums define a **set of fixed values**.

☑ Real-Life Example – Coffee Sizes

```
public class CoffeeOrder {
    enum Size { SMALL, MEDIUM, LARGE }

    public static void main(String[] args) {
        Size myCoffee = Size.MEDIUM;

        System.out.println("You ordered a " + myCoffee + " coffee.");
    }
}
```

📝 **Tip:** Enums are great for limited choices (e.g., days of the week, pizza sizes, etc.)

---

8. Java Keywords

Examples: `int`, `public`, `class`, `return`, `static`, etc. These are **reserved words** and cannot be used as variable names.

---

9. Comments in Java

- Single-line: `// This is a comment`
- Multi-line:

```
/* This is a
   multi-line comment */
```

---

10. Using Blank Lines

Java ignores blank lines and extra spaces. Use blank lines to organize your code and improve readability.

---

11. Inheritance (Intro Only)

Java allows new classes to "inherit" features from other classes.

```java
// Parent class
class Animal {
    void eat() {
        System.out.println("Animal is eating.");
    }
}

// Child class inherits from Animal
class Dog extends Animal {
    void bark() {
        System.out.println("Dog is barking.");
    }
}

public class TestInheritance {
    public static void main(String[] args) {
        Dog myDog = new Dog();
        myDog.eat();  // Inherited method
        myDog.bark(); // Own method
    }
}
```

More on this in future sessions.

---

12. Interfaces (Intro Only)

An interface defines behavior that a class agrees to implement.

```java
// Define an interface
interface Playable {
    void play();
    void stop();
}

// Implement the interface
class MusicPlayer implements Playable {
    public void play() {
        System.out.println("Playing music.");
    }
    public void stop() {
        System.out.println("Music stopped.");
    }
}

public class TestInterface {
    public static void main(String[] args) {
        MusicPlayer player = new MusicPlayer();
        player.play();
        player.stop();
    }
}
```

You'll explore interfaces in depth later on.

Activities

◇ Activity 1: Print Your Info

Task: Display your personal details using simple variables.

```java
public class AboutMe {
    public static void main(String[] args) {
        String name = "Leo";
        int age = 19;
        String hobby = "playing guitar";

        System.out.println("Hi! My name is " + name + ".");
        System.out.println("I'm " + age + " years old.");
        System.out.println("I enjoy " + hobby + ".");
    }
}
```

☑ **Tip:** Declare each variable clearly, and use `+` to concatenate strings.

◇ Activity 2: Favorite Movies (Array)

Task: Store your top 3 favorite movies in an array and print them.

```
public class FavoriteMovies {
    public static void main(String[] args) {
        String[] movies = {"Inception", "Interstellar", "The Matrix"};

        System.out.println("My Favorite Movies:");
        System.out.println(movies[0]);
        System.out.println(movies[1]);
        System.out.println(movies[2]);
    }
}
```

☑ **Tip:** Keep index access simple for now—no need for loops yet.

◇ Activity 3: Enum for Clothing Sizes

Task: Define an enum for clothing sizes and print your preferred size.

```
public class ShirtSize {
    enum Size { SMALL, MEDIUM, LARGE }

    public static void main(String[] args) {
        Size mySize = Size.LARGE;
        System.out.println("My shirt size is: " + mySize);
    }
}
```

☑ **Tip:** Use enums for fixed categories like sizes, levels, or days.